Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Simon Hiti

Sistem za avtomatizacijo upravljanja akvarija

diplomsko delo

univerzitetni študij računalništva in informatike

Mentor: doc. dr. Boštjan Slivnik

Ljubljana 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01974 / 2013
Datum: 5.11.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **SIMON HITI**

Naslov: **SISTEM ZA AVTOMATIZACIJO UPRAVLJANJA AKVARIJA**
SYSTEM FOR AUTOMATIC MANAGMENT OF AN AQUARIUM

Vrsta naloge: DIPLOMSKO DELO UNIVERZITETNEGA ŠTUDIJA

Tematika naloge:

Izdelajte sistem za upravljanje akvarija. Ta sistem naj omogoča avtomatsko doziranje treh vrst gnojil in čiščenje sistema gnojenja, merjenje temperature vode in hlajenje vode ter prižiganje in ugašanje glavne luči akvarija. Sistem naj bo kar se da prilagodljiv: za vsako vrsto gnojila naj bo možno nastaviti količino gnojila, gnojenje pa naj bo možno nastaviti za vsak dan v tednu posebej. Sistem naj omogoča uporabo in spreminjanje nastavitev brez uporabe dodatnega računalnika.

Mentor: B. Slivnik
doc. dr. Boštjan Slivnik

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Simon Hiti, z vpisno številko **63070117**, sem avtor diplomskega dela z naslovom:

Sistem za avtomatizacijo upravljanja akvarija

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Boštjan Slivnik,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 1. februar 2014 Podpis avtorja:

Zahvaljujem se družini za podporo pri izdelavi diplomske naloge in celotnega študija. Zahvaljujem se tudi mentorju za uspešno mentorstvo.

Kazalo

Povzetek

V okviru diplomske naloge smo naredili neodvisen sistem, ki omogoča avtomatizirano gnojenje, hlajenje akvarija in prižiganje luči. Samodejne funkcije olajšajo delo akvaristu in mu prihranijo čas. Sistem temelji na razvojni ploščici Arduino Mega 2560 in njenih komponentah. Za razvoj sistema smo uporabili programski jezik C/C++, ki je prilagojen za Arduino, razvojno okolje Arduino in Visual Studio 2013. Sistem vsebuje štiri črpalke, ki omogočajo gnojenje, rele, ki omogoča prižiganje in ugašanje luči, in temperaturni senzor, ki omogoča vklop računalniškega ventilatorja za hlajenje vode. Vse skupaj nadziramo in nastavljamo prek prikazovalnika in tipkovnice velikosti 4x4.

Ključne besede

gnojilo, dozirna črpalka, avtomatizacija akvarija

Abstract

For the purpose of this diploma paper we have built an independent system which enables automated fertilising, aquarium cooling process and on/off switching of the lights. Automatic functions make the aquarist's work easier and are a great time saver. The system is based on the development microcontroller board Arduino Mega 2560 and its components. For developing the system the programming language C/C++ adjusted to Arduino was used, along with the Arduino development environment and Visual Studio 2013. The system comprises four pumps providing automated fertilising, a relay providing on/off switching of the lights and a temperature sensor providing on switching of the computer ventilator for water cooling. All together is controlled and operated/adjusted on the display and 4x4 keyboard.

Key words

fertiliser, dosing pump, aquarium automation

Poglavje 1

Uvod

Akvaristika je hobi ljudi, ki se ukvarjajo z akvariji in organizmi v njih. Obstaja več vrst akvarijev: rastlinski, morski, skupinski in biotopni. Najboljši akvariji so biotopni, ker popolnoma posnemajo določeno naravno okolje (reko, jezero,...). To pomeni, da rastline, pesek, kamni,... v biotopnem akvariju ustvarjajo tak življenjski prostor, kakršen je v okolju, ki ga akvarij posnema (npr. v Amazonki).

Najbolj nas zanimajo rastlinski akvariji, ki so lahko tudi biotopni ali ne. Poleg rastlin lahko v akvarij vključimo tudi vodne živali – ribe, školjke, rakci,... Ta vrsta akvarija je poimenovana skupinski akvarij.

V naši diplomi obravnavamo nebiotopni skupinski akvarij. Naša naloga je poskrbeti, da imajo naše rastline vsak dan dovolj svetlobe in hranljivih snovi, živali pa primerno temperaturo. Pri osvetlitvi moramo paziti, da imamo dovolj W/l vode; nekatere rastline potrebujejo več svetlobe, druge manj. Svetlobe nikakor ne sme biti preveč oziroma luč ne sme biti predolgo prižgana, sicer se začnejo razvijati alge. Na razvoj alg poleg svetlobe vpliva tudi količina hranljivih snovi oziroma gnojil, ki jih dodajamo v akvarij. Teh ne sme biti preveč niti premalo. Gnojila se delijo na dve podskupini: makro- in mikroelementi. K mikroelementom spadajo predvsem kemijski elementi: baker, bor, mangan, molibden, cink, železo in klor. Vsak od njih ima svojo vlogo. K makroelementom spadajo kemijski elementi: dušik, fosfor, kalij,

kalcij, žveplo in magnezij. Tudi vsak od teh opravlja svojo nalogo.

Včasih navedena gnojila ne zadostujejo za popolno uspevanje rastlin v akvariju, zato moramo kak element izmed zgoraj naštetih dodatno odmeriti: rdečelistne rastline, na primer, potrebujejo železo, da lahko ohranjajo svojo barvo.

Priporočljivo je, da se te snovi dodajo ob istem dnevnem času in da se zagotovijo hranila vsaj eno uro prej, preden se prižgejo luči. Tako se hranila enakomerno premešajo po celotnem akvariju in vse rastline dobijo enako količino potrebnih snovi. Pri ročnem odmerjanju si pomagamo z medicinsko injekcijsko brizgo. Avtomatizacija bi nam pomagala zagotoviti stalno uro odmerjanja in natančneje odmerjeno količino gnojila. Prihranila bi nam tudi nekaj časa v primerjavi z ročnim načinom.

V našem akvariju imamo tudi vodne živali, zato moramo paziti na temperaturo vode, ki mora biti približno taka kot v naravnem okolju, iz katerega prihajajo živali. Če je voda prehladna, moramo uporabiti akvarijski grelnik, da vodo segrejemo do primerne temperature. Problem nastane poleti, ko je ozračje pretoplo in ogreva vodo. Takrat je treba poskrbeti za hlajenje vode oziroma pospešiti izhlapevanje, da se voda hladi. Tudi v tem primeru bi nam zelo koristila avtomatika, s katero bi stalno spremljali temperaturo vode in ob morebitnem povišanju temperature s hlajenjem samodejno vzdrževali idealno temperaturo vode.

Nekaj naprav, ki samodejno opravljajo zgoraj našteta dela, že obstaja na tržišču. Žal moramo za vsako nalogo kupiti ločeno napravo, kar je cenovno neugodno. Poleg tega smo omejeni na uporabniške vmesnike in ne moremo posegati v način delovanja posameznih naprav. Če bi napravo izdelali sami, bi lahko vse te funkcije združili, kar bi bilo ceneje in delovanje sistema bi lahko prilagodili našim potrebam.

Poglavje 2

Sistem za avtomatizacijo upravljanja akvarija

Sistem za avtomatizacijo upravljanja akvarija smo zasnovali tako, da vsebuje štiri črpalke. Predvidoma bi bile tri za gnojila, četrta pa za izpiranje teh iz cevnega sistema (ali vse štiri za gnojila, odvisno od potreb uporabnika). Vključili smo še temperaturni senzor, ventilator za hlajenje in možnost vklopa/izklopa luči. Vse to krmili mikroprocesorski sistem s prikazovalnikom.

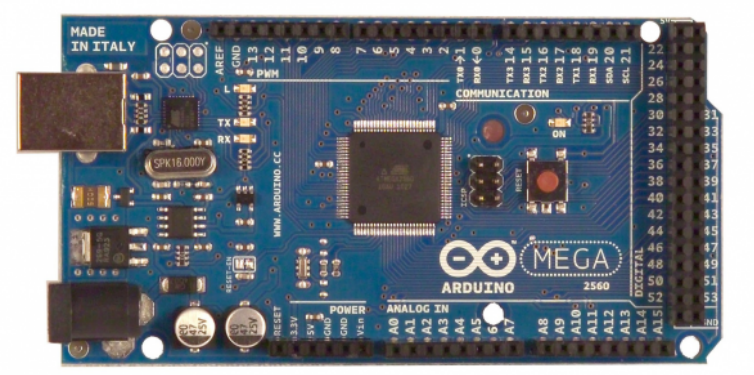
2.1 Uporabljene komponente

Pri našem sistemu za avtomatizacijo upravljanja akvarija smo uporabili naslednje komponente:

- Arduino MEGA 2650 rev3,
- uro realnega časa DS1307,
- temperaturni senzor DS18B20,
- prikazovalnik LCD 2004,
- peristaltične dozirne črpalke,

- tipkovnico 4x4,
- dvokanalni relejni modul,
- napajalni del,
- 12-voltni računalniški ventilator.

2.1.1 Arduino MEGA 2650 rev3



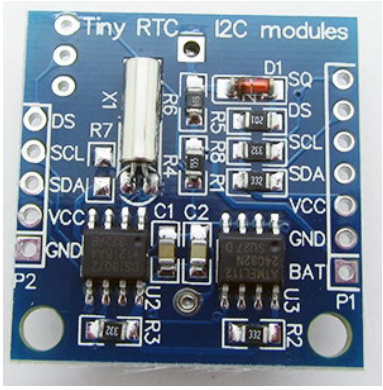
Slika 2.1: Arduino MEGA 2650 rev3[1].

Komponenta Arduino MEGA 2650 rev3 [3] na sliki 2.1 je ena izmed razvojnih ploščic Arduino Mega tretje generacije (v nadaljnjem besedilu: Arduino). Uporablja mikrokrmilnik ATmega2560. Ima 54 digitalnih vhodno-izhodnih pinov, od tega jih ima 15 opcijo pulzno-širinske modulacije. Vsebuje 256kB bliskovnega (ang. flash) pomnilnika in 4 kB EEPROM. To razvojno ploščico smo uporabili zato, ker ima veliko vhodov/izhodov, majhno porabo, enostaven priklop komponent in enostavno programiranje.

2.1.2 Ura realnega časa DS1307 RTC

Uporabili smo tudi modul z uro realnega časa (v nadaljnjem besedilu: realna ura) [4], ki je na sliki 2.2. Vsebuje čip DS1307, na katerem teče realna ura.

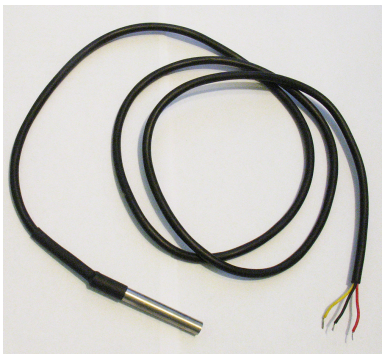
[1]Vir slike: `www.tehnologija.biz/134-thickbox_default/arduino-mega-2560.jpg`



Slika 2.2: Ura realnega časa[2].

Vsebuje tudi baterijo, ki zagotavlja napajanje čipa ob izpadu glavnega vira napetosti. Tako se ura ne izbriše, ampak teče dalje. Ta modul smo uporabili predvsem zato, ker je trenutno edini dostopen na slovenskem trgu.

2.1.3 Temperaturni senzor DS18B20



Slika 2.3: Temperaturni senzor DS18B20.

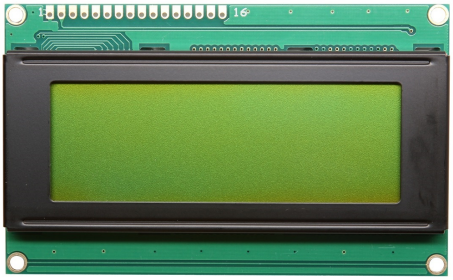
Uporabljamo 8-bitni digitalni temperaturni senzor DS18B20 [5] na sliki 2.3, ki prek enožičnega protokola (ang. one-wire) sprejema ukaze in sporoča svoje stanje oziroma podatke.

[2]Vir slike: `www.tehnologija.biz/279-thickbox_default/rtc-ura-realnega-casa.jpg`

To komponento smo uporabili zato, ker nam omogoča enostavno branje temperature, izražene v dejanski vrednosti. Če bi uporabili analogni temperaturni senzor (termistor), katerega upornost se spreminja glede na temperaturo, bi morali sami izračunati temperaturo glede na trenutno upornost. Prednost pred ostalimi digitalnimi senzorji je v tem, da za izbranega že obstajajo knjižnice za Arduino.

2.1.4 Prikazovalnik LCD 2004

Uporabili smo prikazovalnik LCD 2004 na sliki 2.4, ki ima 4 vrstice in vsaka vsebuje 20 znakov. Z manjšim prikazovalnikom bi bilo težje delati, ker ne bi mogli prikazati vsega, kar želimo. Večji bi bil cenovno dražji, pa tudi za naše ga ne potrebujemo potrebujemo, ker je naš sistem dokaj preprost.



Slika 2.4: LCD velikosti 20x04[3].

2.1.5 Peristaltične dozirne črpalke

Peristaltične dozirne črpalke na sliki 2.5 delujejo na 12 V. Njihova poraba je 0,96 W. Mehanski del je sestavljen iz cevčice, ki jo stiskajo 3 valji, ki jih vrti elektromotor. Ker je cevčica na stiku z valji stisnjena, se tekočina z obratom valjev potiska v želeno smer. Natančnost črpalke je 1 ml/s ±0,02 ml.

Nekaj akvaristov, ki so postavili podoben sistem, je priporočilo te črpalke. So (zadovoljivo) natančne in predvsem delujejo na enosmerno napetost, zato

[3]Vir slike: `http://www.nexuscyber.com/green-backlight-2004a-lcd-module`)

Slika 2.5: Peristaltična dozirna črpalka.

jih lažje krmilimo s pulzno-širinsko modulacijo kot črpalke na izmenično napetost.

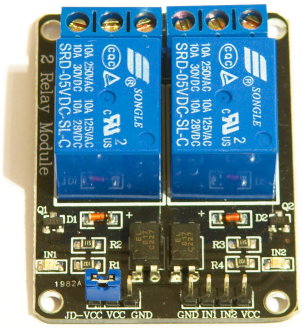
2.1.6 Tipkovnica 4x4

Slika 2.6: Tipkovnica oblike 4x4[4].

Tipkovnica je oblike 4x4, kot vidimo na sliki 2.6. Vsebuje številke od 0 do 9, znaka $*$ in # ter črke A, B, C in D. Priklop je izveden s 4 povezavami za vrstice in s 4 povezavami za stolpce. Ob pritisku na tipko se aktivirata ena vrstica in en stolpec (nastane stik med njima).

[4]Vir slike: `http://www.tehnologija.biz/256-thickbox_default/44-matrix-16-key-membrane-switch-keypad.jpg`

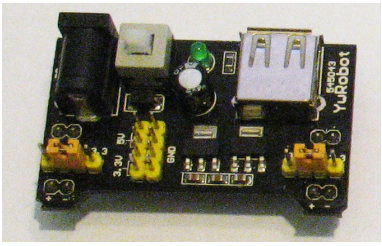
2.1.7 Dvokanalni relejni modul



Slika 2.7: Dvokanalni rele[5].

Dvokanalni relejni modul na sliki 2.7 vsebuje 2 releja, ki ju nadzorujemo prek 2 vhodov. Njuna zgornja meja je 250 V izmenične napetosti (v nadaljnjem besedilu: AC) ali 30 V enosmerne napetosti (v nadaljnjem besedilu: DC). Na modulu poleg relejev je tudi vezje, ki preklaplja releja glede na vhodna stanja.

2.1.8 Napajalni del



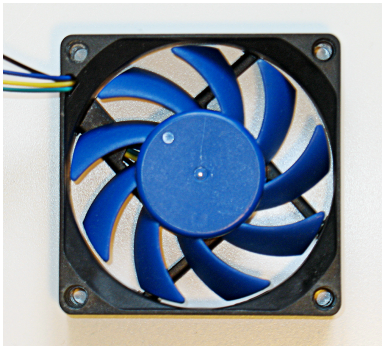
Slika 2.8: Napajalni del.

Za napajalni del smo uporabili 12-voltni (1 A) napajalnik, ker potrebujemo 12 V še za druge komponente. Uporabili smo tudi YwRobot MB102 napajalni modul na sliki 2.8, ki pretvori vhodno napetost v 5 V in/ali 3,3

[5]Vir slike: `http://www.bajdi.com/?attachment_id=938`

V. Vezje je narejeno tako, da ga lahko pritrdimo neposredno na prototipno ploščico (ang. protoboard). To omogoča napajanje neodvisno od Arduina in vir treh različnih napetosti: 12 V, 5 V in 3,3 V. Alternativno bi lahko sistem napajali tudi iz Arduina.

2.1.9 12-voltni računalniški ventilator



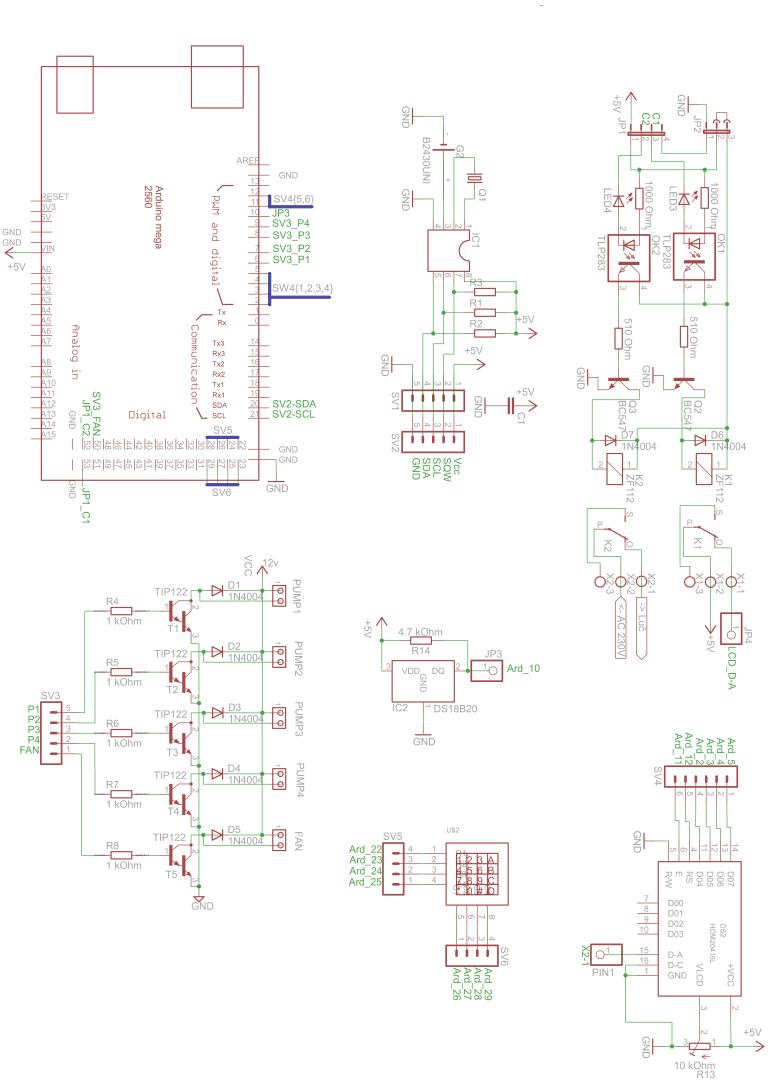
Slika 2.9: Računalniški ventilator.

Uporabili smo tudi 12-voltni računalniški ventilator na sliki 2.9. Ima 4-polni priklop: 12 V, maso, izhodne podatke o hitrosti vrtenja in kontrolo hitrosti vrtenja prek pulzno-širinske modulacije.

2.2 Shema

Na sliki 2.10 je prikazana naša shema. Levo zgoraj je Arduino. Desno od njega sta realna ura in relejni modul. Pod Arduinom je naše sestavljeno vezje, ki omogoča vklop in izklop štirih črpalk in ventilatorja. Desno od vezja je temperaturni senzor in pod njim tipkovnica. Desno od tipkovnice je prikazovalnik.

Na shemi lahko vidimo celotno vezje realne ure. Ker je modul tovarniško izdelan, nas zanimajo le priklopi, označeni s SV2 in SV1. Ker uporablja I2C-komunikacijo, moramo ustrezno povezati SDA- in SCL-pin z našim Arduinom na istoimenska pina. Pin V_{cc} povežemo na 5V, GND pa na skupno maso.



Slika 2.10: Shema vezave komponent.

Na shemi lahko vidimo celotno vezje relejnega modula, ki ravno tako tovarniško izdelan. Zato sta za nas pomembna le priklopa JP2 in JP1, ki se povežeta z Arduinom. Pina 2 in 3 priklopa JP1 lahko povežemo na kateri koli digitalni izhod Arduina. Mi smo si izbrali izhoda 52 in 53, ker nam je zaradi postavitve komponent tako bolje ustrezalo. Na spodnji strani sheme modula imamo priklopa X1 (rele K1) in X2 (rele K2). Sredinska priklopa (X1-2 in X2-2) sta vhoda na releja, kamor priklopimo vhodno napetost naprave, ki jo želimo krmiliti. Izhoda X1-1 in X2-1 sta običajno odprta izhoda (ang. normally-open contacts), X1-3 in X2-3 pa sta običajno zaprta izhoda (ang. normally-closed contacts). Glede na potrebe si izberemo enega izmed izhodov releja in nanj priklopimo napravo.

Vezje za kontrolo črpalk in ventilatorja smo morali sestaviti sami. Omogoča kontrolo enosmernih napetosti višjih od 5 V. Priklop SV3 (leva stran) se poveže z Arduinom. Pri tem moramo SV3 s pini 2, 3, 4 in 5 povezati na pine Arduina prek pulzno-širinske modulacija (v nadaljnjem besedilu: PWM). Uporabili smo pine 6, 7, 8 in 9, ker vsi delujejo na istem časovniku in imajo tako enako izhodno frekvenco. SV3, pin 1 lahko povežemo na običajni digitalni izhod, ker za ventilator ne potrebujemo PWM-ja. Mi smo ga priklopili na pin 52. Na desno stran na priklope PUMP1, PUMP2 in PUMP3 povežemo črpalke. Na priklop FAN pa povežemo ventilator. Pri črpalkah polariteta priklopa ni pomembna, ker z njo le določamo smer črpanja (priporočljiva je ista smer). Pri ventilatorju moramo paziti na smer vrtenja: če se ventilator vrti v napačno smer, zaradi oblike krakov ne bo pričakovanega učinka ventilacije.

Temperaturni senzor (DS18B20) potrebuje 5-voltni priklop in maso. Priklop LP3 povežemo na Arduino na enega izmed digitalnih vhodov/izhodov, ker je senzor digitalen in uporablja digitalno komunikacijo. Mi smo ga priklopili na digitalni vhod/izhod 10.

Za tipkovnico potrebujemo 8 digitalnih vhodov/izhodov. Mi smo si izbrali pine od 22 do 29, zato da imamo vse na enem mestu in zaradi boljše preglednosti povezav/žic.

Pri prikazovalniku imamo dva priklopa PIN2 in SW4. PIN2 se poveže na

rele priklopa X1-1. Ta povezava je anoda za osvetlitev ozadja. Potenciometer R13 omogoča nastavitev kontrasta znakov, ki se izpisujejo na prikazovalnik. R/W-pin omogoča pisanje na zaslon/branje z zaslona. Potrebno je omogočiti le pisanje, zato R/W-pin povežemo na skupno maso. SW4 priklopimo na digitalne izhode Arduina, kot je razvidno na shemi. Ni nujno, da se uporabimo iste izhode. Če uporabimo druge, moramo ustrezno spremeniti program.

2.3 Postopek izdelave

Glavni del celotnega izdelka je Arduino (razdelek 2.1.1). Le ta ima tako digitalne kot analogne vhode/izhode. V nadaljevanju bomo opisali, kako smo nanje priključili ostale komponente.

Prebrali smo specifikacije razvojne ploščice Arduino in nekaj recenzij uporabnikov, ki priporočajo, naj ob priklopu večjega števila naprav na Arduinu raje uporabimo zunanji vir napetosti 5 V in/ali 3,3 V (razdelek 2.1.8). Sam Arduino namreč ne zmore napajati velikega števila naprav. To smo tudi storili in vse naprave napajali prek napajalnega modula.

Za začetek smo povezali med sabo Arduina in tipkovnico (razdelek 2.1.6). Ker tipkovnica deluje kot dvodimenzionalna mreža, smo jo lahko priklopili direktno na Arduino brez kakršnih koli dodatnih uporov in ostalih komponent. Vrstice smo priključili na digitalne vhode/izhode 22, 23, 24 in 25, stolpce pa na 26, 27, 28 in 29. Knjižnica nam sama določi, kateri priključki delujejo kot vhodi in kateri kot izhodi.

Sledil je relejni modul. Tukaj ni nič posebnega: priklop za 5 V, masa in vhodna signala. Paziti moramo le na vhodna signala, ker ima modul obratno logiko: rele preklopi, ko stanje vhoda nastavimo na logično '0' oziroma ko stanje vhoda deluje kot masa. Modul nam omogoča tudi prilagajanje napajanja. Če pustimo mostiček (ang. jumper), bosta isti vir napajanja uporabljala tako logika kot sam rele. Če pa mostiček odstranimo, lahko ločeno napajamo iz dveh različnih virov. Pomembno je, da ne presežemo največje dovoljene napetosti ali da ne pripeljemo premajhne. Tukaj smo en kanal uporabili za

krmiljenje luči in tega povezali na Arduinov izhod 53. Pri tem mora rele krmiliti z izmenično napetostjo (AC) 230 V, da lahko prižiga in ugaša luč.

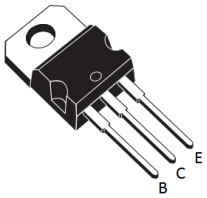
Naslednja komponenta, ki smo jo priklopili, je bil LCD-prikazovalnik (razdelek 2.1.6). Moral smo mu prilotati podnožje ter glede priklopa malo pobrskati po internetu. Ugotovili smo, da njegov priklop ni nič drugačen kot pri ostalih zaslonih manjših velikosti (npr. 16x2). Potrebuje 5-voltno napajanje in maso. Pin V_0 (nekje ga označujejo tudi V_{LCD}) se uporablja za kontrast znakov. Tukaj smo potrebovali potenciometer velikosti 10 kΩ, s katerim lahko nastavljamo kontrast znakov. Potenciometer ima stranska kontakta povezana na 5 V in na maso, srednji kontakt pa na kontakt zaslona V_0. Katodo osvetlitve ozadja smo povezali na maso, anodo pa na rele (razdelek 2.1.7), ker tako lahko ob izklopu zaslona ugasnemo tudi osvetlitev. Vezava prek tranzistorja je bila neuspešna, ker le ta pobere nekaj napetosti in do osvetlitve zaslona je ne pride skoraj nič – zaslon le bledo sveti. Da lahko krmilimo vklop/izklop osvetlitve, smo enega izmed kanalov relejnega modula povezali na Arduinov digitalni izhod 52. Kot lahko vidimo, ima zaslon tudi 8 podatkovnih linij (D_0, ..., D_7). Tukaj smo se lahko odločali med dvema vezavama, to je med 8- ali 4-bitni načinom. Izbrali smo 4-bitni, kar v večini primerov zadošča in tudi povezav/žic je manj.

Ker pri našem projektu potrebujemo točno uro tudi ob izpadu elektrike, smo uporabili realno uro (razdelek 2.1.2). To smo samo priklopili na povezavi SCL (linija ure) in SDA (podatkovna linija), ki sta del podatkovne linije I2C.

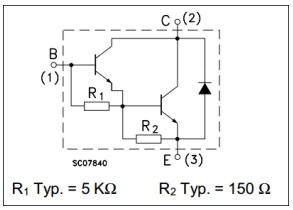
Za detekcijo temperature smo morali še priklopiti ustrezen senzor (razdelek 2.1.3). Pozitivno žico smo priklopili na 5 V in negativno na maso. Osrednjo komunikacijsko žico pa smo morali priklopiti na dvigovalni upor (ang. pullup resistor) in na Arduinov digitalni vhod/izhod 10.

Nazadnje smo morali še priklopiti črpalke (razdelek 2.1.5) in ventilator (razdelek 2.1.9). Oboje potrebuje enosmerni vir napetosti 12 V. Arduino takega izhoda nima. Zato smo se morali prilagoditi. Poiskali smo po internetu in našli nekaj vezalnih shem. Črpalke črpajo le v eno smer in ventilator se vrti le v eno smer, zato ne potrebujemo krmiljenja prek H-mostiča. Vezava

prek relejev ni prišla v poštev, ker pri črpalkah uporabljamo PWM. Na voljo smo imeli tudi vezavo prek tranzistorjev. Izbrali smo darlingtonove NPN-tranzistorje tipa TIP122 (obstajata tudi TIP120 in TIP121). Bolje bi bilo, da bi uporabili unipolarne tranzistorje MOSFET, ker so hitrejši in imajo manjšo porabo, a bi jih morali kupiti. Tranzistorje, ki smo jih že imeli na voljo, smo povezali tako, da smo baze (B) tranzistorjev vezali prek uporov 1kΩ na Arduinove digitalne izhode. Vzpostavili smo krmiljenje tranzistorjev in preostalo nam je le še, da nanje priklopimo črpalke. Pozitivni kontakte črpalk smo priklopili na 12 V, negativne pa na kolektorje (C) tranzistorjev. Emitorje (E) tranzistorjev smo povezali na skupno maso. Ker pa se vse skupaj krmili s pomočjo PWM, je bilo priporočljivo obratno vezati diode tipa 1N4004 med kontakte črpalk (da diode gledajo od mase proti izhodu 12 V). Diode omogočajo glajenje špic ob vklopu in/ali preklopih črpalk, da ne prihaja do motenj na tranzistorjih. Za računalniški ventilator velja enako, le da tukaj nismo uporabili PWM, ker se nam je zdela nepotrebna.

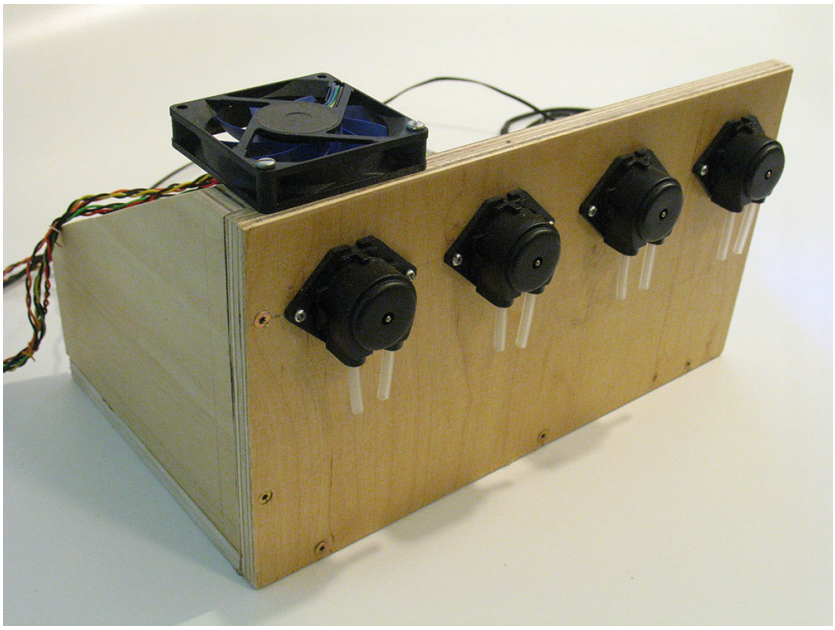


(a) TIP122.

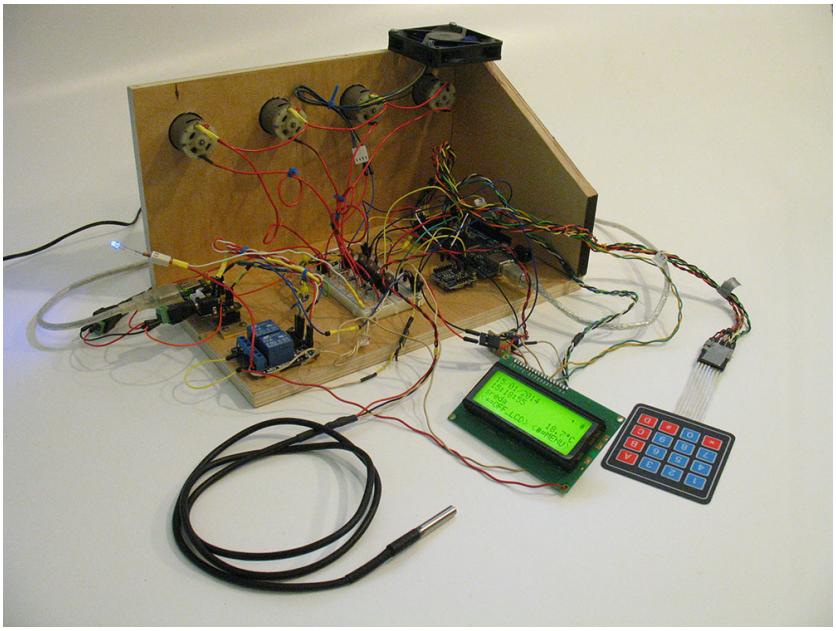


(b) Shema TIP122.

Slika 2.11: Darlingtonov tranzistor TIP122[6].



Slika 2.12: Sprednja stran sistema.



Slika 2.13: Notranjost sistema.

Poglavje 3

Krmiljenje sistema

V prejšnjem poglavju smo opisali, kako smo priklopili posamezne komponente na Arduina. Potrebno je bilo še napisati program za mikroprocesor. Arduino je odprtokoden, zato so knjižnice za komponente prosto dostopne. Nekaj jih ponuja že istoimensko podjetje, druge pa napišejo zvesti uporabniki.

Najprej smo potrebovali razvojno okolje. Na začetku smo programirali v originalnem Arduino razvojnem okolju, ki ga ponuja istoimensko podjetje. Ker pa je zelo neprilagodljiv za velike projekte (nima samodopolnjevanja, pregleda nad uporabljenimi spremenljivkami in možnimi ukazi,...), smo se odločili za uporabo Microsoft Visual Studio 2013 skupaj z dodatkom VisualMicro. Slednji je dostopen skupaj z navodili na `http://playground.arduino.cc/Code/VisualMicro`. Zaradi knjižnic in prevajalnika je treba imeti naloženo prvotno razvojno okolje.

Razvija se ga lahko v čistem C-ju ali v C/C++. Odločili smo se za C/C++, ker je programiranje v C/C++ nekoliko lažje in ni nekaterih omejitev kot pri C-ju. Kot smo že omenili, smo pri programiranju uporabili knjižnice, ki smo jih v program vključili z ukazom #Include<>. Arduinov program se razlikuje od navadnega le v tem, da nima funkcije *main*(), ampak funkciji *setup*() in *loop*(). Funkcija *setup*() se izvede le ob zagonu programa, funkcija *loop*() pa se izvaja neprekinjeno v nekakšni neskončni zanki.

Koda 3.1: Goli program – osnovni funkciji.

```
void setup()
{ //zagonska koda }
void loop()
{ //ponavljajoča se koda }
```

Pri pisanju programa je potrebno paziti, da so spremenljivke, ki morajo ohraniti svojo vrednost, definirane kot globalne spremenljivke, drugače se jim ob vsakem novem obhodu funkcije *loop*() pobriše vrednost. Znotraj te funkcije tudi izvajamo ukaze in kličemo ostale funkcije.

Za začetek smo priklopili realno uro. Knjižnica in primeri za delo z njo so dostopni na `https://github.com/adafruit/RTClib/`. Ta knjižnica nam omogoča, da enostavno le preberemo vrednosti iz realne ure ali jih zapišemo nanjo. Pri tem ni treba da vemo kako realna ura deluje, ker za to v ozadju skrbi knjižnica. Mi le iz nje pokličemo ukaz in mu podamo pripadajoče parametre.

Koda 3.2: Priprava realne ure za uporabo v programu.

```
#include <Wire.h>
#include "RTClib.h"
RTC_DS1307 rtc; //objekt na uro
void setup()
{
  Wire.begin(); //zagon I2C komunikacije
  rtc.begin();
}
void loop()
{ //... }
```

Ker smo želeli vzpostaviti neodvisen sistem od računalnika, smo morali priklopiti še prikazovalnik in tipkovnico. Knjižnica za prikazovalnik je vključena že v uradnem razvojnem okolju, za tipkovnico pa je posebej dostopna na uradni strani Arduino `http://playground.arduino.cc/Main/KeypadTutorial`. Prikazovalnik smo najprej definirali (naredili zanj objekt), nato smo izvedli inicializacijo prikazovalnika: določili velikost (16x2, 20x4,...),

št. menija	meni
-1	ugasnjen zaslon
0	prikaz ure, datuma, temperature
1	izbira nastavitev
2	nastavitev ure in datuma
3	izbira črpalke
4	nastavitev temperature
5	nastavitev črpalke$_x$
6	izbira dnevov delovanja črpalke$_x$
7	nastavitev ure za vklop/izklop luči
10	potrditev nastavitev ure in datuma
11	potrditev nastavitev črpalke$_x$
12	potrditev nastavitev temperature ali luči

Tabela 3.1: Vsi meniji na zaslonu in pripadajoče identifikacijske številke.

na katerih kontaktih ima priključene podatkovne linije in na katerih 'reset' in 'enable'. Nato smo ga lahko začeli uporabljati oziroma pisati nanj.

LCD zna prikazati le osnovne ascii znake. Posebne znake je potrebno ročno izrisati. Na prikazovalniku prikazujemo različne menije. Vsak je označen z identifikacijsko številko, kot je prikazano v tabeli 3.1.

Pri tipkovnici smo najprej definirali dvodimenzionalno tabelo znakov (vsaki tipki določili pripadajoči znak), nato določili pine, na katere so priklopljene vrstice in stolpci (vsak tipka ima svojo pozicijo). Ob pritisku na tipko nam razred vrne eno od definiranih vrednosti. Tukaj smo naredili funkcijo 'keyDetect(key)', ki nam zazna, katera tipka je bila pritisnjena, in nam na podlagi menija, v katerem se nahajamo, izvede ustrezno operacijo/funkcijo. Prav tako smo naredili funkcijo 'displayPrint()', ki zagotavlja izpisovanje podatkov na prikazovalnik.

Koda 3.3: Del kode, ki zaznava tipke in izpisuje na zaslon.

```
#include <LiquidCrystal.h>
#include <Keypad.h>
//...
//##### LCD #####
//lcd pins: rst, e, d0, d1, d2, d3
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
int lcdBaclightPin = 52; //izhod za osvetlitev lcd
byte cetrtek[8] = { //črka č
  B01010,
  B00100,
  B01110,
  B10001,
  B10000,
  B10001,
  B01110,
  B00000,
};
//##### KEYPAD #####
//znaki tipkovnice - definirane vrednosti
char keys[4][4] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
//kontakti na Arduinu
byte rowPins[ROWS] = { 22, 23, 24, 25 };
byte colPins[COLS] = { 26, 27, 28, 29 };
//definicija objekta keypad
Keypad kpd = Keypad( makeKeymap(keys), rowPins, colPins,
                     4, 4);
//spremenljivke
int meni = 0;
outDispl = -1; //določa enkraten izpis na lcd
//## funkcije ##
void displayPrint()
```

```
{
  //izpis na zaslon
  DateTime now;
  String dan_v_tednu;
  int j = 0;
  switch(menu)
  {
    case -1:
      if (outDispl != -1)
      {
        outDispl = -1;
        lcd.noBlink();
        //digital.write(lcdBaclightPin, LOW)
      }
      break;

    case 0: //screenSaver, clock, date
      now = rtc.now();
      if(outDispl != 0)
      {
        if(outDispl == -1)
          digitalWrite(lcdBaclightPin, HIGH);
        lcd.clear();
        lcd.setCursor(0, 3);
        lcd.print("<*=OFF_LCD> <#=MENU>");
        outDispl = 0;
        //lcd.noCursor();
        lcd.noBlink();
      }
      lcd.setCursor(0, 0);
      if(now.day()<10)
        lcd.print("0");
      lcd.print(now.day(), DEC);
      lcd.print('/');
      if(now.month()<10)
        lcd.print("0");
      lcd.print(now.month(), DEC);
      lcd.print('/');
```

```
      lcd.print(now.year(), DEC);
      lcd.setCursor(17, 0);
      if (temperatureState)
        lcd.write(byte(5));
      else
        lcd.write(byte(2));
      lcd.setCursor(19, 0);
      if (lightState)
        lcd.write(byte(3));
      else
        lcd.write(byte(2));
      lcd.setCursor(0, 1);
      if(now.hour()<10)
        lcd.print("0");
      lcd.print(now.hour(), DEC);
      lcd.print(':');
      if(now.minute()<10)
        lcd.print("0");
      lcd.print(now.minute(), DEC);
      lcd.print(':');
      if(now.second()<10)
        lcd.print("0");
      lcd.print(now.second(), DEC);
      lcd.setCursor(0, 2);
      dan_v_tednu = danVTednu(now.dayOfWeek());
      if (dan_v_tednu == "cetrtek")
      {
        lcd.write(byte(0));
        lcd.print("etrtek");
      }
      else
        lcd.print(dan_v_tednu);
      lcd.setCursor(14, 2);
      temperature = temperatureRead();
      lcd.print(temperature);
      lcd.setCursor(18, 2);
      lcd.write(byte(4));
      lcd.print("C");
```

```
      break;
    //case ...
  }
}
void keyDetect(char key)
{
  //detekcija pritisnjene tipke
  switch(key):
  {
    case '*':
      switch(meni):
      {
        case 0:
          //izklop zaslona
          lcd.NoDisplay();
          //izklop osvetlitve LCDja
          digitalWrite(lcdBaclightPin, LOW);
          meni = -1;
          break;
        case 1:
          meni = 0;
          break;
        case 2:
          meni = 1;
          break;
        //...
    case '#':
      switch(meni):
      {
        case -1:
          //vklop zaslona
          lcd.display();
          //vklop osvetlitve LCDja
          digitalWrite(lcdBaclightPin, HIGH);
          meni = 0;
          break;
        case 2:
          e1 = pravilnost_ure();
```

```
          e2 = pravilnost_datuma();
          if(e1 == 0 && e2 == 0)
          {
            meni = 10;
            //zapis ure na RTC modul
            rtc.adjust(DATE, TIME);
          }
          else
          {
            //izpis napake
            meni = 2;
          }
          break;
        //...
      }
    //... A, B, C, D
    default:
      //detekcija števil (0...9)
      lcd.print(key);
      break;
  }
}
void setup()
{
  //znak č shranimo pod byte(0)
  lcd.createChar(0, cetrtek);
  lcd.begin(20, 4); //definiranje velikosti zaslona
  pinMode(lcdBaclightPin, OUTPUT); //LCD BL
}
void loop()
{
  char key = kpd.getKey();
  displayPrint();
  keydetect(key);
}
```

Sedaj lahko izpisujemo menije na zaslon, se premikamo med njimi in vnašamo vrednosti (uro, datum, količino gnojila v mililitrih, čas gnojenja,...) ter jih

nato potrdimo in shranimo. Zato ima vsaka tipka svojo funkcijo. Števila služijo za vnašanje vrednosti, zvezdica (∗) kot izhod iz menija oziroma prekinitev vnosa, lojtra (#) kot potrditev nastavitve ali vklop prikazovalnika in črke (A, B, C, D) kot funkcijske tipke (izbira menija, premik na vrstico, vklop/izklop podsistema,...).

Vsaka nastavitev mora imeti v ozadju svojo logiko, ki zagotavlja, da so vrednosti pravilno vnesene. Pri vnosu nove ure je treba preverjati pravilnost vnosa ure (ure, minute, sekunde) in datuma (dan, mesec, leto). Modul realne ure ima shranjen le koledar od leta 2000 naprej in to za nadaljnjih 100 let (do 2099). Za naše potrebe je to dovolj, ker ne potrebujemo sistema, ki bi deloval večno. Uro je treba še pretvoriti v format, ki ga pozna modul realne ure, in pretvorjeno uro zapisati oziroma shraniti nanj: zapis 'DD/MM/LLLL' (npr. 23. 1. 2014) smo moral pretvoriti v 'mmm DD LLLL' (npr. Jan 23 2014), pri čemer so meseci v angleščini. Ure ni bilo treba pretvarjati, ker je že bila v primernem formatu 'UU:MM:SS'. Pri tem smo uporabili tipki A in B, ki služita za premik med vrsticama, ter tipko D za premik kurzorja za 1 mesto nazaj.

Pri vnosu vrednosti za črpalke je bilo ravno tako treba preverjati uro. Urediti smo morali še vnos količine gnojila v mililitrih na desetinko natančno (od 0,0 ml do 99,9 ml). Da pa logika ve, kateri dan mora poskrbeti za doziranje določenega gnojila, smo napisali meni, v katerem lahko izberemo posamezne dni v tednu ali vse dni skupaj. Vse nastavitve črpalk se shranijo v integriran EEPROM na Arduinu. Ob ponovnem zagonu programa se te nastavitve preberejo. Ko imamo vnesene nastavitve, sistem preverja (zaporedoma od 1. do 4. črpalke) v enosekundnih intervalih, ali je katera črpalka nastavljena za tekoči dan in ali je nastavljeni čas enak trenutnemu. V ozadju imamo narejeno čakalno vrsto: tako lahko ena izmed črpalk izvaja gnojenje, druge pa jo čakajo, da konča. Preverjanje časa se izvaja tudi, ko neka črpalka deluje, ker smo uporabili "zakasnitev brez zakasnitve"ali z drugimi besedami: razlika v milisekundah med trenutnim časom in časom vklopa črpalke mora biti večja, da se črpalka izklopi. Tako tudi dosežemo zakasnitev 1 sekunde.

Tukaj imajo tipke A, B in D enako funkcijo kot pri vnosu ure. Dodali pa smo še tipko C, ki nam odpre meni za določanje dni delovanja črpalke.

Koda 3.4: Krmiljenje črpalk.

```
//...
long pumpPreviousMillis = 0; //ms
interval = 1000; //ms
void doziranjeGnojil()
{
  int pwm = 250;
  DateTime now;
  unsigned long currentMillis = millis();

  if (currentMillis - pumpPreviousMillis > interval)
  {
    //pregledovanje časov črpalk
    String x[2];
    timeToString(x);
    int tmpmax=0;
    now = rtc.now();
    int tmpday = now.dayOfWeek() - 1;
    pumpPreviousMillis = currentMillis;
    //pregled zadnjega mesta v vrsti
    for (int i = 0; i < 4; i++)
      if (pumpReady[1][i]> tmpmax)
        tmpmax = pumpReady[1][i];
    //dodajanje črpalk v čakalno vrsto
    for (int i = 0; i < 4; i++)
    {
      if (pumpaDaysPrew[i][tmpday] == 1)
        if (x[0] == pumpaClock[i])
        {
          ++tmpmax;
          pumpReady[1][i] = tmpmax;
        }
    }
  }
  currentMillis = millis();
```

```
  //če nobena črpalka ne deluje, potem vklopimo
  if (pumpReady[0][0] == 0 && pumpReady[0][1] == 0 &&
     pumpReady[0][2] == 0 && pumpReady[0][3] == 0)
  {
    for (int i = 0; i < 4; i++)
    {
      if (pumpReady[1][i] == 1)
      {
        pumpReady[0][i] = 1;
        PumpRunningTmp = i;
        analogWrite(pumpPin[i], pwm);
        pumpPreviousMillisR = currentMillis;
        break;
      }
      else
        PumpRunningTmp = -1;
    }
  }
  //spremljamo, katera črpalka deluje in kdaj je napočil čas
     za njen izklop
  else
  {
    currentMillis = millis();
    if (currentMillis - pumpPreviousMillisR > pumpMililiters[
       PumpRunningTmp] * 1000)
    {
      pumpPreviousMillisR = currentMillis;
      pumpReady[0][PumpRunningTmp] = 0;
      for (int k = 0; k < 4; k++)
        if (pumpReady[1][k] != 0)
          pumpReady[1][k]--;
      analogWrite(pumpPin[PumpRunningTmp], 0);
    }
  }
}
```

Naš sistem tudi vsebuje tudi logiko za vklop luči. Tukaj moramo vnesti dve uri - za vklop in izklop luči. Oba vnosa je potrebno preveriti. Omogočeno je

tudi aktiviranje/deaktiviranje (tipka D), odvisno od tega, ali želimo to logiko uporabljati ali ne. In seveda imamo na voljo ročni način vklopa/izklopa luči (tipka C). Logika, ki v ozadju skrbi za vklop/izklop, deluje enako kot pri črpalkah: enkrat na sekundo preverja trenuten čas s shranjenim, in ko pride do enakosti, izvede operacijo vklopa/izklopa. Na podlagi stanja luči se tudi odloča, kateri čas primerja. Tukaj tipki A in B opravljata enako funkcijo kot pri predhodnih nastavitvah. Tipki C in D pa dobita funkciji, ki smo ju že predhodno omenili. Naj še omenimo, da v osnovnem meniju tipka C omogoča ročni način vklopa/izklopa luči.

Sedaj nam preostane le še logika, ki skrbi za temperaturo. Tukaj imamo funkcijo, ki omogoča branje temperature iz samega senzorja (to počne v enosekundnih intervalih). Ta vrednost se nato izpisuje na prikazovalniku poleg ure. V meniju za vnos temperature nastavimo največjo in najmanjšo dovoljeno temperaturo. Pri vnosu vrednosti imamo omejitev navzgor 40 °C in navzdol 15 °C, in seveda pazimo, da vrednosti nista vneseni v obratnem vrstnem redu. Če sta vrednosti enaki, je logika narejena tako, da gleda odstopanje za ±0,5 °C od nastavljene temperature. Tukaj imamo prav tako možnost aktiviranja/deaktiviranja logike (tipka D). Naloga te logike je, da vklopi sistem, ko temperatura vode preseže največjo dovoljeno vrednost in da izklopi, ko temperatura pade pod najnižji dovoljeni prag. A to le v primeru, ko je ta del sistema v samodejnem in ne v ročnem načinu. Temperaturo vode očitava v intervalu 1 sekunde, ker smo omejeni s samim senzorjem, ki potrebuje med ukazom za pretvorbo podatkov in ukazom za branjem le teh iz registra vsaj 750 ms. Tipki A in B imata enako vlogo kot pri ostalih nastavitvah. Tipka C omogoča ročni vklop/izklop ventilatorja. V osnovnem meniju pa ročni vklop/izklop lahko izvedemo s tipko D.

Koda 3.5: Nadzor temperature in vklop/izklop računalniškega ventilatorja.

```
//...
long temperaturePrevFan = 0;
int temperatureInterval = 1000;
void ventilatorAuto()
{
```

```
unsigned long currentMillis = millis();
if (currentMillis - temperaturePrevFan >
   temperatureInterval)
{
  temperaturePrevFan = currentMillis;
  if (temperatureMin < temperatureMax)
  {
    if (temperature > temperatureMax)
    {
      //vklop ventilatorja
      digitalWrite(temperatureFanPin, HIGH);
      temperatureState = 1;
    }
    else if (temperature < temperatureMin)
    {
      //izklop ventilatorja
      digitalWrite(temperatureFanPin, LOW);
      temperatureState = 0;
    }
  }
  else if (temperatureMin == temperatureMax)
  {
    if (temperature > temperatureMax + 0.5)
    {
      digitalWrite(temperatureFanPin, HIGH);
      temperatureState = 1;
    }
    else if (temperature < temperatureMin - 0.5)
    {
      digitalWrite(temperatureFanPin, LOW);
      temperatureState = 0;
    }
  }
  else
  {
    //napaka pri vnosu temperature
    temperatureEnable = 0;
  }
```

```
  }
}
```

Vse vrednosti, ki jih vnašamo, se shranjujejo v začasne spremenljivke in se šele ob potrditvi nastavitev prenesejo v glavne spremenljivke. To pa zato, ker če sami vnašamo neko nastavitev, in se vmes odločimo prekiniti vnos, mora glavna spremenljivka še vedno vsebovati prvotno vrednost.

Poglavje 4

Testiranje

Kot vemo, mora naš sistem dozirati natančno količino gnojil. Za to smo morali najprej testirati črpalke, kako velik je največji pretok skozi njih. To smo naredili tako, da smo podstavili na vhodno stran poln kozarec vode, na izhodno pa prazen (še prej smo izmerili njegovo težo). Črpalko smo pognali za 10 s in izmerili novo težo predhodno praznega kozarčka in to delili z 10, da smo dobili ml/s (1 g $\doteq$ 1 ml), Prišli smo do zaključka, da črpalka prečrpa približno 1 ml tekočine v 1 s pri nastavljenem PWM-ju na 250 (maksimalna vrednost je 255). Odstopanje pri tem je približno 0,02 ml.

Slika 4.1: Testiranje pretoka črpalk.

Slika 4.2: Meritev pretoka (1 g $\doteq$ 1 ml).

Celoten sistem smo testirali tako, da smo vnesli vrednosti komponent (ura, dnevi, količina,...). Nato pa smo nastavili uro nekaj minut pred prvim dogodkom. Ko so se vsi dogodki (črpalke, luč) uspešno izvedli, smo dobili potrditev, da stvar deluje.

Prikazovalnik v povezavi s tipkovnico smo testirali sproti, ko smo sestavljali menije. Le tako smo lahko dosegli, da se med meniji premikamo pravilno, da vnos deluje pravilno in da se kurzor premika pravilno. Preverjati smo morali tudi, ali se vrednosti pravilno shranjujejo. To smo storili tako, da smo jih izpisovali prek serijskih vrat na računalnik. Pravilno smo morali izpisati tudi opozorila pri vnosu napačnih vrednosti.

Pri realni uri smo uporabili primer programa, ki je bil poleg knjižnice. Ko nam je program izpisal pravilne vrednosti prek serijskih vrat, smo vedeli, da je pravilno priklopljen. Ko smo napisali kodo za naš program, nam je ta moral na prikazovalnik izpisati prave vrednosti.

Temperaturni senzor smo testirali tako, da smo z njega prebrali vrednosti in jih izpisovali na prikazovalnik. Poleg njega smo postavili drugi tempera-

turni senzor (npr. vremensko postajo). Najprej smo ju postavili na sobno temperaturo enega poleg drugega in nato še v hladno vodo. Obakrat sta kazala enake vrednosti (z manjšim odstopanjem ±0,2 °C). To pomeni, da je naš senzor deloval pravilno.

Na koncu, ko smo uspešno stestirali vse komponente in smo vedeli, da sistem deluje pravilno, smo ga priklopili na akvarij, kot je razvidno na sliki 4.3.

Slika 4.3: Sistem priklopljen na akvarij.

Poglavje 5

Zaključek

Naša rešitev še ni popolna. Lahko bi dodelali še nekaj podrobnosti. Problem je pri modulu realne ure. Ko je bil ta napajan neposredno iz Arduina (ta pa prek USB-ja), je deloval pravilno. Sedaj pa, ko sta oba priklopljena na napajalni del, ga je občasno ob zagonu potrebno izklopiti in znova priklopiti, da nato deluje pravilno. Popraviti bi morali tudi izpis na prikazovalnik. Ob zagonu je potrebno Arduino resetirati, da pravilno izpisuje znake na prikazovalnik. Morda pa je to napaka na samem Arduinu.

Sistem morda vsebuje še kakšno napako, ki je nismo uspeli odkriti med testiranjem.

Za nadaljnje delo bi sistem nadgradili tako, da bi spremenili strukturo strojne opreme, zamenjali bi darlingtonove trazistorje z MOSFET, ker imajo manjšo porabo in so hitrejši (še posebej ker uporabljamo PWM). Dodali bi še modul z relejem, na katerega bi priklopili grelnik. Nato pa bi spremenili nastavitve temperature, pri kateri bi le nastavili idealno/želeno temperaturo in dovoljeno odstopanje. Ko bi temperatura presegla mejno vrednost, bi se prižgal ventilator, ko pa bi padla pod želeno, bi se prižgal grelnik. Ko bi temperatura dosegla želeno vrednost, bi se izvedel izklop ene izmed naprav (odvisno od tega, katera bi bila takrat v uporabi). Mogoče bi bilo dodati še senzor gladine vode skupaj z vodno črpalko. Senzor bi nadzoroval, koliko vode izhlapi in ob določenem mejnem pragu vklopil črpalko, ki bi dodala

vodo v akvarij in dvignila gladino vode na ustrezno višino.

Literatura

[1] B. W. Evans, "Arduino programing notebook", (2007). Dostopno na: http://playground.arduino.cc/uploads/Main/arduino_notebook_v1-1.pdf (oktober 2013)

[2] Arduino examples. Dostopno na: http://arduino.cc/en/Tutorial/HomePage (oktober 2013)

[3] Arduino mega 2560 rev3. Dostopno na: http://arduino.cc/en/Main/ArduinoBoardMega2560 (oktober 2013)

[4] Ura realnega časa DS1307. Dostopno na: http://www.alldatasheet.com/datasheet-pdf/pdf/254791/MAXIM/DS1307.html (november 2103)

[5] Digitalni temperaturni senzor DS18B20. Dostopno na: http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf (december 2013)